A Study of the interaction of BGP/OSPF in Zebra/ZebOS/Quagga

Avinash Ramanath
avinash_ramanath@hotmail.com

ABSTRACT
Border Gateway Protocol (BGP) allows an autonomous system to maintain connectivity with other autonomous systems. This report is a study of this protocol and its implementation in the Zebra [6] /ZebOS Server [7] /Quagga routing software [10]. The software architecture of this routing software is studied along with details of the implementation of the interaction between BGP and Open Shortest Path First (OSPF).

Keywords
Border Gateway Protocol, Open Shortest Path First, Zebra, ZebOS, Internet Routing.

1. INTRODUCTION

Border Gateway Protocol (BGP) is an inter-autonomous system routing protocol. An autonomous system (AS) is a network or a group of networks under a common administration and with common routing policies. BGP is the current Internet standard [1], for inter-domain (AS) exterior routing. The primary function of a BGP speaking system is to exchange reachability information with other nodes running BGP. This information essentially contains the routing information to reach an AS or a set of ASs. This information is sufficient to construct the AS connectivity from which routing loops may be pruned and policy decisions enforced at the AS level. BGP runs over TCP, a reliable transport protocol, for establishing connections and eliminates the need for retransmission, acknowledgement and sequencing of packets.

An AS governs a set of routers/nodes under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASs. Any node/router running BGP as the routing protocol is referred to as BGP speaker or BGP system. BGP speakers connected within an AS constitute “Internal peers”, and BGP speakers connected across ASs constitute “External Peers”.

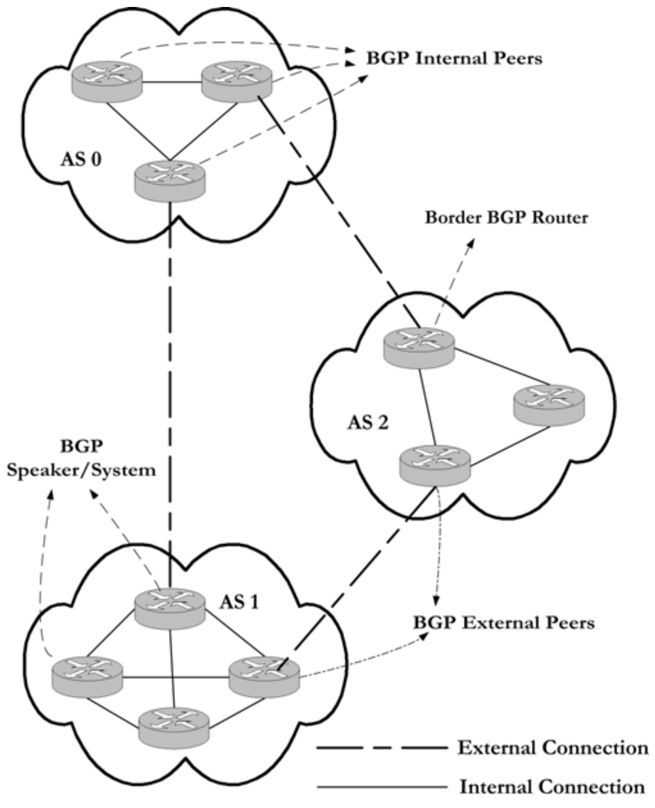


Fig. 1

Fig. 1 illustrates the terminology associated with BGP. ASs 0, 1, and 2, each contain a set of routers/nodes running BGP, thus manifesting as BGP Speakers/Systems. As depicted in the diagram, any router/node which maintains a connection external to its AS, is referred to as a Border Router. The routers/nodes which form part of external connections are referred to as BGP External peers, as indicated by the connections existing between border routers in AS 1 and AS 2. The routers/nodes within a single AS which maintain a meshed topology are referred to as BGP Internal peers, as indicated by routers present in each of the ASs.

2. BGP Protocol Basics

Two systems running BGP form a transport protocol connection, exchange messages and confirm the connection parameters. After the connection establishment, the entire routing tables at each of the BGP running nodes are transferred to each other. Later on with changes in topology, resulting in changes to routing tables, incremental updates are exchanged between the BGP systems. KEEPALIVE messages are sent periodically to ensure the aliveness of the connection. Notification messages are sent in response to errors or special conditions. If a connection encounters an error condition, a notification message is sent and the connection is closed.

Route advertisements are carried out between BGP peers in UPDATE messages, with the destination represented by the reachability information, and the path is the information stored in the path attributes of this message. Routes are stored in Routing Information Bases (RIBs), namely Adj-RIB-In, Loc-RIB, and Adj-RIB-Out. Routes received from other BGP speakers are stored in Adj-RIB-In. Routes that will be used by the local BGP speaker must be present in the Loc-RIB, and the forwarding information base (FIB) stores the next hop information for the routes. Routes that will be advertised to other BGP speakers are saved in Adj-RIB-Out. The routes, which are selected for installing in the Loc-RIB, are decided by invoking a Decision process as illustrated in the next sub-section.

BGP uses the following attributes for qualifying the routes that are used for routing as well as those that are advertised on to neighbors:

1. ORIGIN:

This attribute defines the origin of the path information.

2. AS_PATH:

This attribute identifies the autonomous systems through which routing information carried in an UPDATE message has passed.

3. NEXT_HOP:

The NEXT_HOP path attribute defines the IP address of the border router that should be used as the next hop to the destinations listed in an UPDATE message.

4. MULTI_EXIT_DISC:

This attribute may be used on external (inter-AS) links to discriminate among multiple exit or entry points to the same neighboring AS.

5. LOCAL_PREF:

This attribute is included in all UPDATE messages that are advertised to BGP speakers/systems within the advertising BGP speaker's AS. A BGP speaker calculates the degree of preference for each external route and includes it in the LOCAL_PREF attribute when advertising a route to its internal peers.

6. ATOMIC_AGGREGATE:

Whenever a BGP speaker, presented with a set of overlapping routes (routes with a common prefix) from one of its peers, selects the less specific route without selecting the more specific one, then the BGP system attaches the ATOMIC_AGGREGATE attribute to the route when propagating it to other BGP speakers.

7. AGGREGATOR:

This attribute which informs about a BGP speaker performing route aggregation, is included in UPDATE messages.

2.1. Decision Process

Decision process is used for installing routes at the local router/node and selecting routes for subsequent advertisement by applying policies to the incoming routing information stored in Adj-RIB-In. The output of this process is the set of routes to be advertised to all peers being stored in Adj-RIB-Out, and the set of routes that are used by the local BGP speaker, for forwarding packets, to be stored in Loc-RIB.

The first phase of the decision process determines the degree of preference for the route, as LOCAL_PREF or preconfigured policy information, if the route is from the local autonomous system. If the route belongs to a non-local autonomous system, the degree of preference is determined using preconfigured policy information.

The second phase of the decision process involves the selection of the routes feasible for addition to the Loc-RIB. This phase selects the route with either the highest degree of preference, or a single route to a destination. In the case of a tie between various routes to the same destination, the following criteria are applied iteratively for breaking the tie:

a. Select the route with the lowest MULTI_EXIT_DISC attribute
b. Select the route with the lowest cost (interior distance).
c. If there are several routes with the same cost:
    a) Select the route advertised by a BGP speaker in a neighboring AS, with the lowest ID
    b) Otherwise, select the route advertised by a BGP speaker whose ID is the lowest.

The third phase of the decision process involves the process of route dissemination, whenever one of the following conditions occurs:

a. When Loc-RIB has changed
b. When locally generated routes learnt by means outside of BGP has changed

c. When a new BGP speaker-BGP speaker connection is established.

2.2. Packet traversal in a BGP domain

BGP provides a networked view of ASs, with various ASs sending traffic to other ASs. The below provides an insight into the traversal of a packet across ASs.

Fig 1. provides an illustration of a network topology with different ASs.

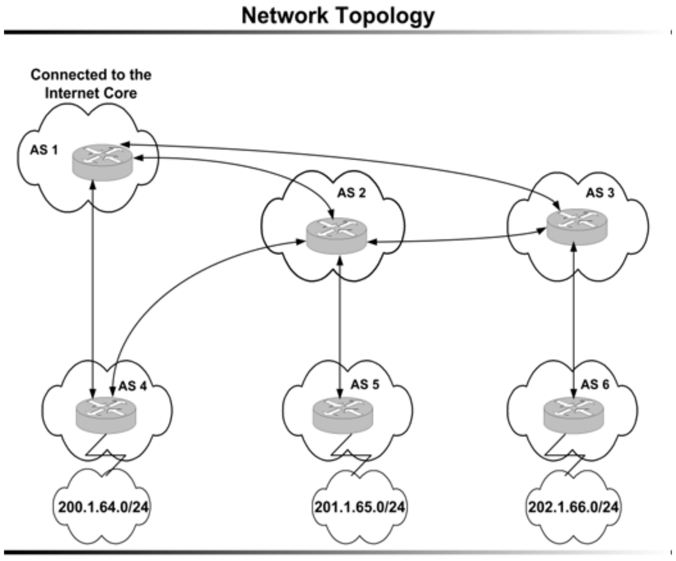
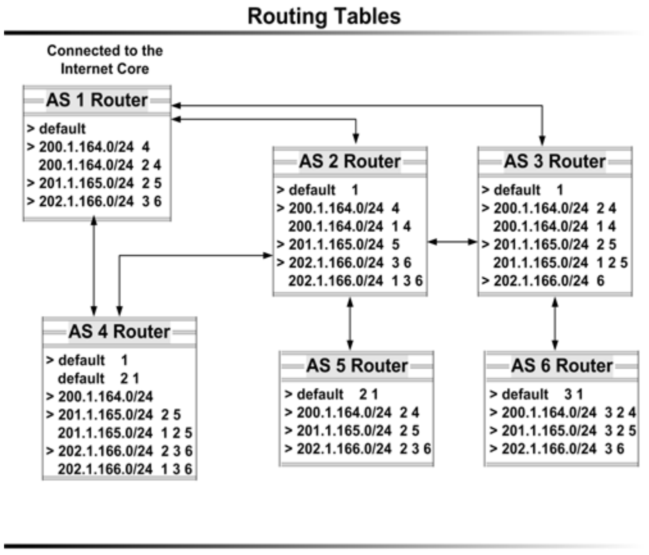


Fig. 2

For clear understanding of the BGP routing process, each of the ASs is assumed to include single routers. AS1 is connected directly to the Internet core, thus providing connectivity to the outer networked world. ASs 2 and 3 provide transit service to other ASs which need connection to the outer world, and are thus referred to as *Transit AS*. ASs 4, 5 and 6 provide networking connections to the networks 200.1.164.0/24, 201.1.165.0/24, 202.1.166.0/24 respectively.



```
AS 1 Router
> default
> 200.1.164.0/24  4
  200.1.164.0/24  2 4
> 201.1.165.0/24  2 5
> 202.1.166.0/24  3 6
```

```
AS 2 Router
> default    1
> 200.1.164.0/24  4
  200.1.164.0/24  1 4
> 201.1.165.0/24  5
> 202.1.166.0/24  3 6
  202.1.166.0/24  1 3 6
```

```
AS 3 Router
> default    1
> 200.1.164.0/24  2 4
  200.1.164.0/24  1 4
> 201.1.165.0/24  2 5
  201.1.165.0/24  1 2 5
> 202.1.166.0/24  6
```

```
AS 4 Router
> default    1
  default    2 1
> 200.1.164.0/24
> 201.1.165.0/24  2 5
  201.1.165.0/24  1 2 5
> 202.1.166.0/24  2 3 6
  202.1.166.0/24  1 3 6
```

```
AS 5 Router
> default    2 1
> 200.1.164.0/24  2 4
> 201.1.165.0/24  2 5
> 202.1.166.0/24  2 3 6
```

```
AS 6 Router
> default    3 1
> 200.1.164.0/24  3 2 4
> 201.1.165.0/24  3 2 5
> 202.1.166.0/24  3 6
```

Fig. 3

The routing tables at each of the BGP routers in the ASs are illustrated in Fig. 3. Whenever there are alternate routes to reach a network, the BGP Decision Process sets up a preferred route (as is indicated by the symbol ">" in front of the route). The BGP routers install a "***default***" route to forward the packets, whose destination network is not listed in the routing table, to a default router. The numbers next to the destination network address refers to the AS-path to be followed to reach the destination network.

The process of the routing table creation for reaching the network 200.1.164.0/24 is illustrated below:

i. AS 4's BGP routers advertises, to its neighbors, the network information of 200.1.0/24 network attached to it.

ii. AS 1's and AS 2's BGP routers install this routing information in their routing tables, and advertise the same to its neighbors.

iii. AS 1's and AS 2's BGP routers do not advertise the above received information to AS 4, as it is the originator of the routing information.

iv. The advertisement from AS 1 and AS 2 BGP routers reach AS 3 and AS 5, and AS3 advertises this information on to AS 6. In this way all the BGP routers in all ASs receive the routing information.

v. In the event when a BGP router receives more than one route to a particular destination network, the BGP Decision Process decides the route to be installed as per policy decisions. This event occurs at the BGP router present in AS 3, which receives advertisements from AS 1 and AS 2 to reach 200.1.164.0/24.

3. Zebra/ZebOS/Quagga routing suite

Zebra [6] / ZebOS Server Routing Suite [7]/ Quagga [10], provide for distributed multi-server multi-threaded routing facilities. Zebra/Quagga are open source software. Zebra software is commercially produced as ZebOS Server Routing Suite, and Quagga is a fork of the Zebra software. All these software share the same architecture for managing the various protocol daemons. Traditional routing software (such as GateD [5]) is made as one process program that provides all of the routing protocol functionalities as a whole. Zebra and associated software provide for an architecture in which individual protocols function as independent daemons communicating with a single Zebra daemon.

Zebra with IPv4 support can be run on GNU/Linux 2.2.x and 2.4.x, FreeBSD 4.x/5.x, NetBSD 1.6.x, OpenBSD 3.x, and with IPv6 support on FreeBSD, NetBSD, OpenBSD, and GNU/Linux. ZebOS Server Routing Suite can be run on Red Hat Linux 7.x/8.0 with Linux kernel 2.4.x, Solaris 2.8, and FreeBSD 4.4 – 4.6. Quagga can be run on GNU/Linux 2.2.x, 2.3.x, 2.4.x, FreeBSD 2.2.8/3.x/4.x, NetBSD 1.4, OpenBSD 2.5, and Solaris 2.6/7. Support for BGP-4 protocol [1] and RIPv1 [2], RIPv2 [3] and OSPFv2 [4] is extended in the above routing software.

3.1 Architecture

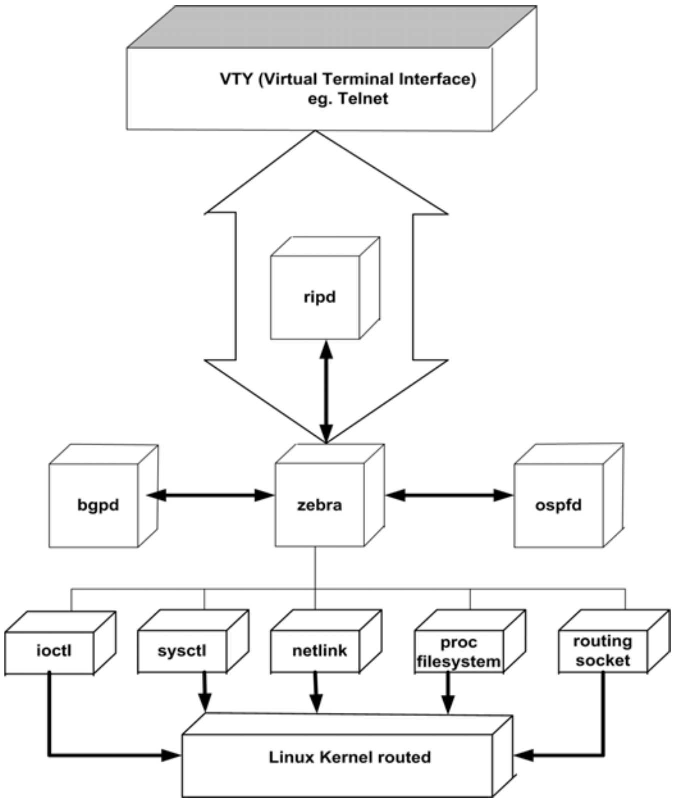


Fig. 4

Fig. 4 [11] illustrates the architecture of the Zebra Routing Daemon, along with the associated daemons of BGP [1], OSPF [4], RIP, RIPv2 [3], RIPng [8] and OSPF6 [9], in Zebra/ZebOS Server Routing Suite software. The Protocol daemons interact with the kernel routing table using Zebra daemon as the intermediary. Zebra defines its own TCP-based protocol to handle inter-process communication between the Zebra daemon and the protocol daemons. Each protocol daemon sends selected routes to Zebra daemon, which is responsible for interacting, and managing the routes to be installed in the forwarding table.

Zebra daemon effectively serves as a moderator for allocation and distribution of services and resources to the various protocol daemons. Each daemon has its own routing table. Zebra daemon maintains the kernel routing table, and is also responsible for redistributing information between the various routing protocol daemons.

3.2. Zebra Daemon

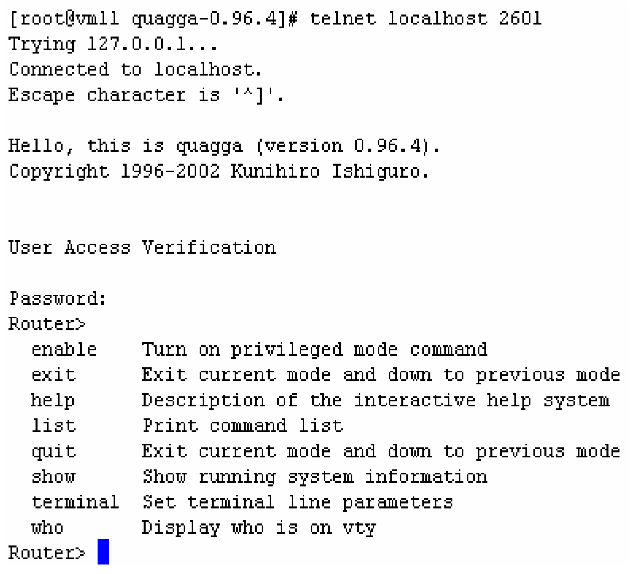
The various routing processes in the routing software [6] [7] run as daemons. The basic feature of a daemon is that it runs in the background and is independent of control from all terminals. All the routing protocols running as daemons can be controlled by the use of a virtual terminal access provided by a telnet-based interface. The daemons can be configured to run on ports different than the default ports. The following is the list of default ports, (as specified in the Zebra routing software [6] installation) on which the various daemons run:

zebra	2601/tcp	# Zebra vty
ripd	2602/tcp	# RIPd vty
ripngd	2603/tcp	# RIPngd vty
ospfd	2604/tcp	# OSPFd vty
bgpd	2605/tcp	# BGPd vty
ospf6d	2606/tcp	# OSPF6d vty

The vty interface provides for different types of nodes for viewing the Command Line Interface (CLI) of a protocol daemon. The CLI serves to specify the commands which can be executed to modify the state of a routing protocol. The general nodes available are:

a) auth_node:

Used for providing the password to gain access through the vty interface to Zebra daemon. Fig. 5 illustrates the commands available after entering the password:



```
[root@vml1 quagga-0.96.4]# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is quagga (version 0.96.4).
Copyright 1996-2002 Kunihiro Ishiguro.


User Access Verification

Password:
Router>
  enable    Turn on privileged mode command
  exit      Exit current mode and down to previous mode
  help      Description of the interactive help system
  list      Print command list
  quit      Exit current mode and down to previous mode
  show      Show running system information
  terminal  Set terminal line parameters
  who       Display who is on vty
Router>
```

Fig. 5

b) view_node:

Used for observing the states of the different routing protocols.

c) auth_enable_node:

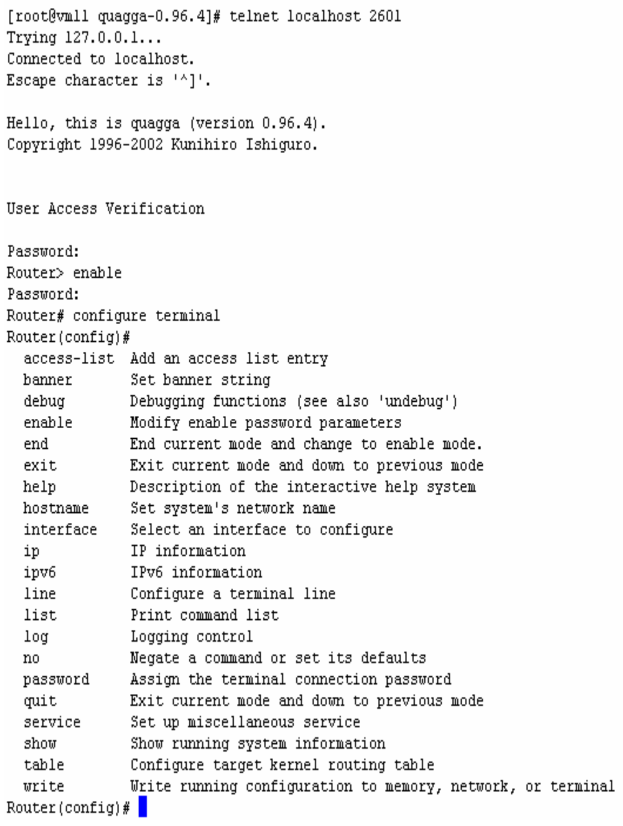
Used for entering the privileged mode by entering an administrative password. Privileged mode provides for configuring routing protocols, debugging the various functions being invoked, and copying the running configuration to the disk.

d) enable_node:

Used for entering the privileged mode without having to enter a password.

e) config_node:

Used to provide the various commands for configuring the routing protocols. Fig. 6 illustrates the commands available in the config_node for configuring the protocols.



```
[root@vml1 quagga-0.96.4]# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is quagga (version 0.96.4).
Copyright 1996-2002 Kunihiro Ishiguro.


User Access Verification

Password:
Router> enable
Password:
Router# configure terminal
Router(config)#
  access-list  Add an access list entry
  banner       Set banner string
  debug        Debugging functions (see also 'undebug')
  enable       Modify enable password parameters
  end          End current mode and change to enable mode.
  exit         Exit current mode and down to previous mode
  help         Description of the interactive help system
  hostname     Set system's network name
  interface    Select an interface to configure
  ip           IP information
  ipv6         IPv6 information
  line         Configure a terminal line
  list         Print command list
  log          Logging control
  no           Negate a command or set its defaults
  password     Assign the terminal connection password
  quit         Exit current mode and down to previous mode
  service      Set up miscellaneous service
  show         Show running system information
  table        Configure target kernel routing table
  write        Write running configuration to memory, network, or terminal
Router(config)#
```

Fig. 6

3.3. BGP Daemon

BGP can be configured by performing a telnet to the particular port on which the BGP daemon vty is running. On gaining access to the vty for the BGP daemon, the user can observe the routing tables in the various daemons, and the running configuration for the routers. For eg, “show ip route” is the command for observing the kernel routing table, and “show ip bgp/ospf/route/interface” is the command for observing the routing tables of the various routing protocols. Initially the configuration runs in the ***view_node*** node. The user will have to get into the ***enable_node/auth_enable_node/config_node*** node for configuring the routers. The interface option is used for determining the IP addresses of the interfaces.

For the purposes of setting up the network ZebOS Server Routing Suite [7] has been used. The following are the steps to configure the BGP routing daemon (present in ZebOS/sbin directory):

1. vtysh #log on to the vty mode for configuring the daemons
2. enable #modifies the mode from view mode to enable mode
3. configure terminal #configures the terminal – used for configuring the routers
4. router bgp <AS No.> #this sets up a BGP router belonging to the <AS No.>

From now on the BGP router can be configured for all the options suitable for configuring BGP protocol daemon. The following are the important commands which have been used for setting up the network:

a) *network A.B.C.D/M:* This command is used for enabling advertisement of this network from this BGP router. This is an important command for enabling routing of a particular network.

b) *neighbor <peer> remote-as <AS No.>:* This command is used for specifying the adjacent neighbor for this BGP router. The parameters are the IP address and the AS number of the peer.

4. Simulation Network and Configuration

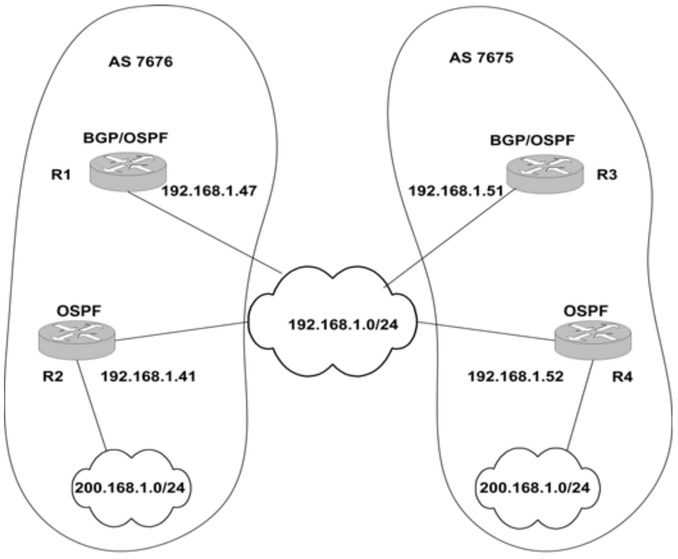


Fig. 7. Network Topology

Fig.7 illustrates the network that has been developed using the Zebra routing software. Two autonomous systems, bearing the numbers 7675 and 7676 have been constructed for the purposes of testing the interaction between the protocols of BGP and OSPF. The main idea behind the simulation is to test/analyze the method BGP running border routers use to communicate redistributed OSPF routes to each other. The next sections (Section 4.1-4.3) illustrate the steps involved in setting up OSPF and BGP routing in the ASs.

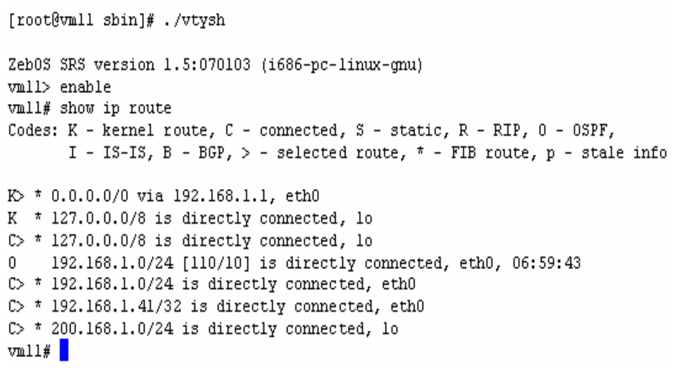
AS 7676 consists of two routers, R1 and R2. R1 provides an interface to the 192.168.1.0/24 network, shared among all the routers in both the ASs, with an IP address of 192.168.1.47. R2 provides two interfaces: one connected to the shared network having an IP address of 192.168.1.41, and another connected to the 200.168.1.0/24 network. AS 7675 consists of two routers, R3 and R4, each of which is connected to the shared network of 192.168.1.0/24. R3 provides a single interface with an IP address of 192.168.1.51. R4 provides two interfaces: one connected to the shared network 192.168.1.0/24 having an IP address of 192.168.1.52, and the other connected to the 203.168.1.0/24 network.

For the purposes of simulation of this network, the 200.168.1.0/24 and 203.168.1.0/24 networks were attached to the loop back interface addresses of routers R2 and R4 respectively. The networks which have been added to the interfaces are brought to the knowledge of the OSPF routing process by redistributing interface information into the OSPF domain. This step had also been included for illustrative purposes. The following commands are used for adding the 200.168.1.0/24 and 203.168.1.0/24 networks to R2 and R4 loop back interfaces, respectively:

R2:

```
> vtysh
> enable
# configure terminal
(config) # interface lo
(config) # ip address 200.168.1.0/24
(config) # router ospf
(config) # ospf router-id 192.168.1.41
(config) # redistribute connected
(config) # neighbor 192.168.1.47
(config) # exit
```

The routing table installed in the router R2 is illustrated in Fig. 8.



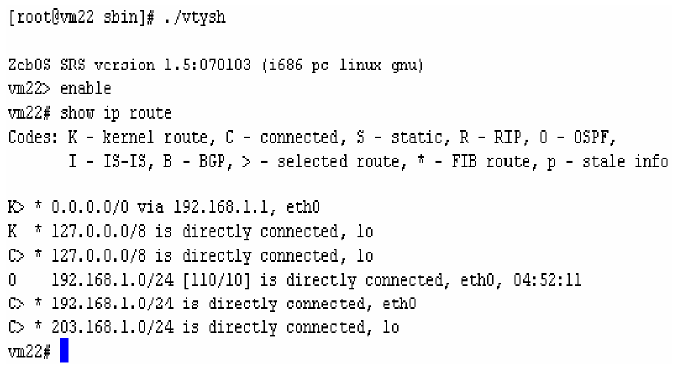
```
[root@vm11 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm11> enable
vm11# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 06:59:43
C> * 192.168.1.0/24 is directly connected, eth0
C> * 192.168.1.41/32 is directly connected, eth0
C> * 200.168.1.0/24 is directly connected, lo
vm11#
```

Fig. 8

R4:

```
> vtysh
> enable
# configure terminal
(config) # interface lo
(config) # ip address 203.168.1.0/24
(config) # router ospf
(config) # ospf router-id 192.168.1.52
(config) # redistribute connected
(config) # neighbor 192.168.1.51
(config) # exit
```

The routing table installed at the router R4 is illustrated in Fig. 9



```
[root@vm22 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686 pc linux gnu)
vm22> enable
vm22# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 04:52:11
C> * 192.168.1.0/24 is directly connected, eth0
C> * 203.168.1.0/24 is directly connected, lo
vm22#
```

Fig. 9

4.1. OSPF Configuration

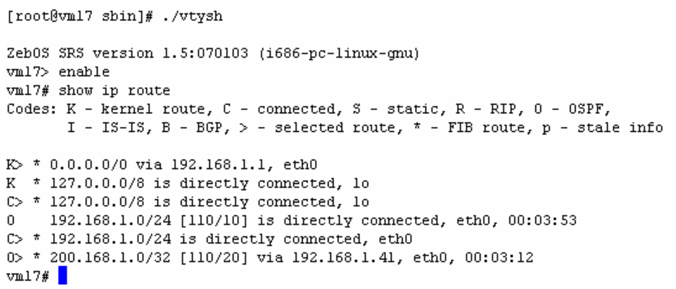
OSPF routing, enabled in all of the routers in the individual ASs, provides for intra-AS routing facility. Now that the routers R2 and R4 are ready with all the networks connected, OSPF routing is configured and enabled in each of the routers R1, R2, R3 and R4 by executing the following commands:

R1:

```
> vtysh
> enable
# configure terminal
(config) # router ospf
```

```
(config) # ospf router-id 192.168.1.47
(config) # area 1 stub
(config) # area 1 default-cost 100
(config) # network 192.168.1.32/27 area 1
(config) # exit
```

The routing table at the router R1 is depicted by Fig. 10.



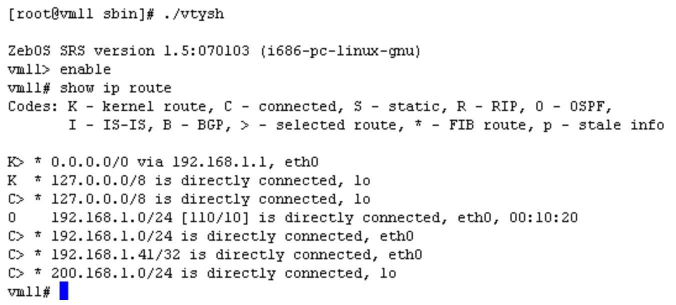
```
[root@vm17 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm17> enable
vm17# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 00:03:53
C> * 192.168.1.0/24 is directly connected, eth0
O> * 200.168.1.0/32 [110/20] via 192.168.1.41, eth0, 00:03:12
vm17#
```

Fig. 10

R2:

```
> vtysh
> enable
# configure terminal
(config) # router ospf
(config) # ospf router-id 192.168.1.41
(config) # area 1 stub
(config) # network 192.168.1.41/32 area 1
(config) # network 200.168.1.0/24   area 1
(config) # exit
```

The routing table installed at the router R2 is depicted in Fig. 11.



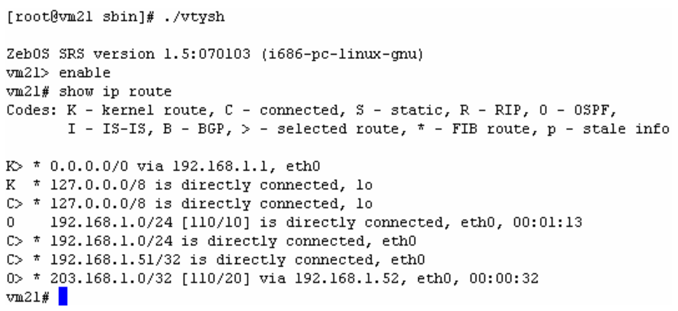
```
[root@vm11 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm11> enable
vm11# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 00:10:20
C> * 192.168.1.0/24 is directly connected, eth0
C> * 192.168.1.41/32 is directly connected, eth0
C> * 200.168.1.0/24 is directly connected, lo
vm11#
```

Fig. 11

R3:

```
> vtysh
> enable
# configure terminal
(config) # router ospf
(config) # ospf router-id 192.168.1.51
(config) # area 2 stub
(config) # area 2 default-cost 100
(config) # network 192.168.1.48/28 area 2
(config) # exit
```

The routing table installed at the router R3 is depicted in Fig. 12.



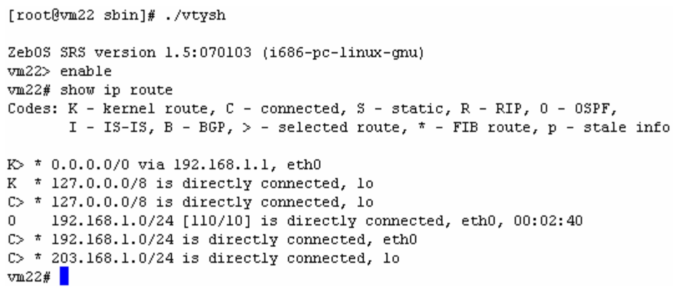
```
[root@vm21 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm21> enable
vm21# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 00:01:13
C> * 192.168.1.0/24 is directly connected, eth0
C> * 192.168.1.51/32 is directly connected, eth0
O> * 203.168.1.0/32 [110/20] via 192.168.1.52, eth0, 00:00:32
vm21#
```

Fig. 12

R4:

```
> vtysh
> enable
# configure terminal
(config) # router ospf
(config) # ospf router-id 192.168.1.52
(config) # area 2 stub
(config) # network 192.168.1.48/28 area 2
(config) # network 203.168.1.0/24   area 2
(config) # exit
```

The routing table installed at the router R2 is depicted in Fig. 13.



```
[root@vm22 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm22> enable
vm22# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - IS-IS, B - BGP, > - selected route, * - FIB route, p - stale info

K> * 0.0.0.0/0 via 192.168.1.1, eth0
K  * 127.0.0.0/8 is directly connected, lo
C> * 127.0.0.0/8 is directly connected, lo
O    192.168.1.0/24 [110/10] is directly connected, eth0, 00:02:40
C> * 192.168.1.0/24 is directly connected, eth0
C> * 203.168.1.0/24 is directly connected, lo
vm22#
```

Fig. 13

4.2. BGP Configuration

BGP routing has to be enabled for facilitating inter-AS routing. BGP has been enabled only in the routers R1 and R3. The following commands are used for enabling BGP on routers R1, and R3:

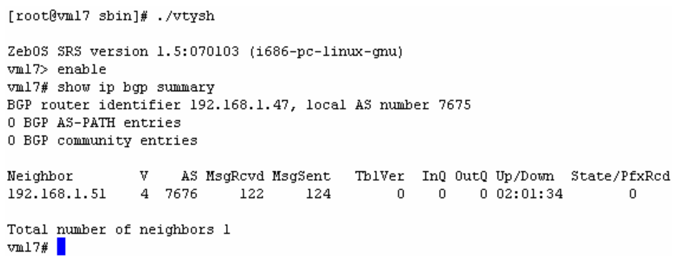
R1:

```
> vtysh
> enable
# configure terminal
(config) # router BGP 7676
(config) # neighbor 192.168.1.51 remote-as 7675
(config) # neighbor 192.168.1.51 interface eth0
```

R3:

```
> vtysh
> enable
# configure terminal
(config) # router BGP 7675
(config) # neighbor 192.168.1.47 remote-as 7675
(config) # neighbor 192.168.1.47 interface eth0
```

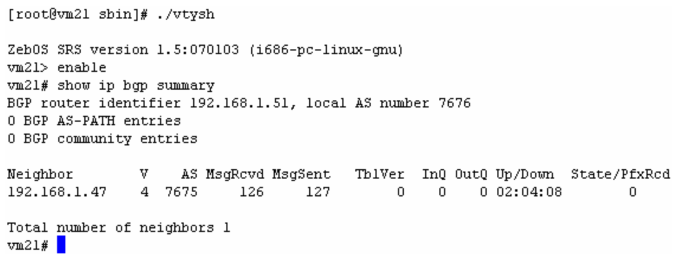
With this configuration, the routers R1 and R3 initiate a TCP connection and start exchanging routing updates. The routing table present at each of the routers R1 and R3 is illustrated in the below section.



```
[root@vm17 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm17> enable
vm17# show ip bgp summary
BGP router identifier 192.168.1.47, local AS number 7675
0 BGP AS-PATH entries
0 BGP community entries

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.1.51    4  7676     122     124        0    0    0 02:01:34        0

Total number of neighbors 1
vm17#
```

Fig. 14

Fig. 14 illustrates the routing table maintained by BGP running at router R1.



```
[root@vm21 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm21> enable
vm21# show ip bgp summary
BGP router identifier 192.168.1.51, local AS number 7676
0 BGP AS-PATH entries
0 BGP community entries

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.1.47    4  7675     126     127        0    0    0 02:04:08        0

Total number of neighbors 1
vm21#
```

Fig. 15

Fig. 15 illustrates the routing table maintained by BGP running at router R3.

4.3. Route Redistribution

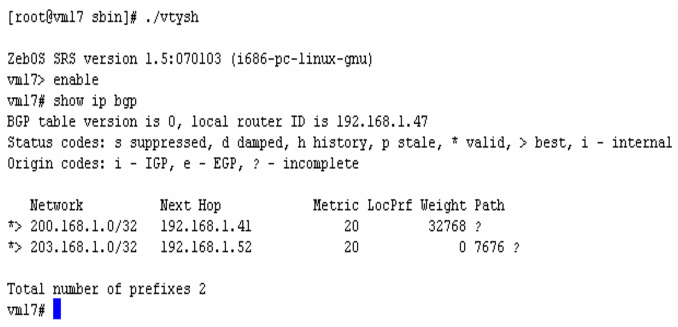
In order to route data packets from R1 and R3 to the networks 200.168.1.0/24 and 203.168.1.0/24, routing information has to be retrieved from OSPF, as OSPF is managing the routing information to these networks. The following commands are used for redistributing the OSPF information into BGP:

R1:

```
(config) # redistribute ospf
(config) # exit
```

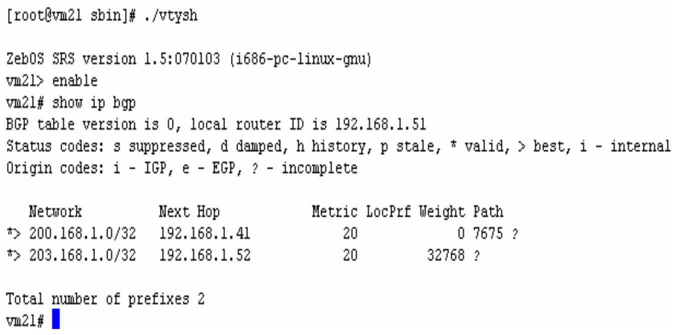
R3:

```
(config) # redistribute ospf
(config) # exit
```



```
[root@vm17 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm17> enable
vm17# show ip bgp
BGP table version is 0, local router ID is 192.168.1.47
Status codes: s suppressed, d damped, h history, p stale, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 200.168.1.0/32   192.168.1.41            20         32768 ?
*> 203.168.1.0/32   192.168.1.52            20             0 7676 ?

Total number of prefixes 2
vm17#
```

Fig. 16



```
[root@vm21 sbin]# ./vtysh

ZebOS SRS version 1.5:070103 (i686-pc-linux-gnu)
vm21> enable
vm21# show ip bgp
BGP table version is 0, local router ID is 192.168.1.51
Status codes: s suppressed, d damped, h history, p stale, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 200.168.1.0/32   192.168.1.41            20             0 7675 ?
*> 203.168.1.0/32   192.168.1.52            20         32768 ?

Total number of prefixes 2
vm21#
```

Fig. 17

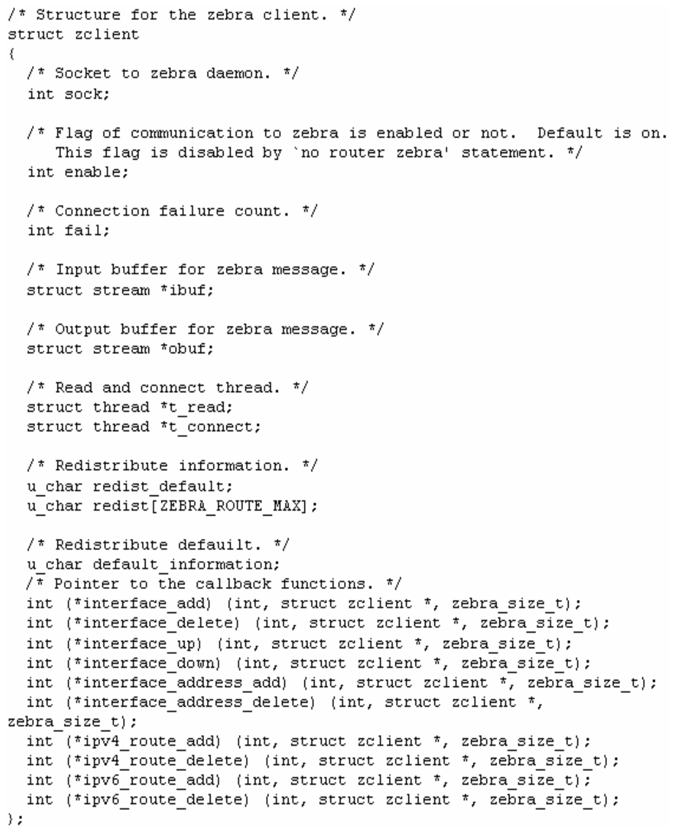
Fig. 16 and fig. 17 illustrate the routing tables at each of the routers R1 and R3, respectively, with redistribution of routes from OSPF. With the above set of commands, the BGP running on R1 and R3 retrieve information about the networks 200.168.1.0/24 and 203.168.1.0/24 respectively. This information is then exchanged between R1 and R3, and each of these routers obtains information about the networks. Router R1 gets to know about the 203.168.1.0/24 network present in the adjacent AS 7675 and router R3 gets to know about the 200.168.1.0/24 network present in adjacent AS 7676. Thus, the redistribution from OSPF on to BGP has been propagated successfully to the adjacent AS.

5. Analysis of the BGP Daemon

The BGP daemon process has been included in the bgpd/ directory of the Zebra 0.93b distribution [6]. The main files of interest for analysis of the code are:

1) bgpd/bgp_vty.c
2) bgpd/bgp_zebra.c
3) zebra/redistribute.c
4) zebra/zserv.c

Whenever a BGP daemon is initialized, it is associated with a set of structures for maintaining information about all the adjacent routers, metrics for the routes, routing tables for each address family and a lot of other related structures. One component of the BGP daemon which is of interest is the "***redist***" array. Most of the arrays associated with the daemon are two dimensional with the first dimension representing the "Address family" (AF_INET, AF_INET6 etc) and the second dimension representing the "Type" (IPv4, IPv6 etc).



```
/* Structure for the zebra client. */
struct zclient
{
  /* Socket to zebra daemon. */
  int sock;

  /* Flag of communication to zebra is enabled or not.  Default is on.
     This flag is disabled by `no router zebra' statement. */
  int enable;

  /* Connection failure count. */
  int fail;

  /* Input buffer for zebra message. */
  struct stream *ibuf;

  /* Output buffer for zebra message. */
  struct stream *obuf;

  /* Read and connect thread. */
  struct thread *t_read;
  struct thread *t_connect;

  /* Redistribute information. */
  u_char redist_default;
  u_char redist[ZEBRA_ROUTE_MAX];

  /* Redistribute defauilt. */
  u_char default_information;
  /* Pointer to the callback functions. */
  int (*interface_add) (int, struct zclient *, zebra_size_t);
  int (*interface_delete) (int, struct zclient *, zebra_size_t);
  int (*interface_up) (int, struct zclient *, zebra_size_t);
  int (*interface_down) (int, struct zclient *, zebra_size_t);
  int (*interface_address_add) (int, struct zclient *, zebra_size_t);
  int (*interface_address_delete) (int, struct zclient *,
zebra_size_t);
  int (*ipv4_route_add) (int, struct zclient *, zebra_size_t);
  int (*ipv4_route_delete) (int, struct zclient *, zebra_size_t);
  int (*ipv6_route_add) (int, struct zclient *, zebra_size_t);
  int (*ipv6_route_delete) (int, struct zclient *, zebra_size_t);
};
```

Fig. 18 Definition of "***struct zclient***"

Fig. 18 depicts the definition of "***struct zclient***" (lib/zclient.h: 35-75) which maintains protocol-specific information for each of the protocol daemons. Information such as the port on which the protocol daemon communicates with the Zebra daemon, input and output stream buffers for maintaining two-way communication with the Zebra daemon, mapping of function pointers to perform actions specific to the functions invoked on the daemon (for eg., ipv4_route_add( ) function pointer to specify the function which is to be invoked whenever an IPv4 route has to be added).

The daemons are also associated with an input stream meant for configuring/modifying the daemon through the vty connection, and an output stream meant for output. For the purposes of the analysis only IPv4 type belonging to the AF_INET address family was used.

a. When the vty connection to the BGP daemon is started, and the user gets into the ***enable*** mode, and starts to configure the daemon, he has the option of redistributing the routes obtained by OSPF. When the "***redistribute ospf***" command is typed into the vty (as illustrated in Section 4), bgp_vty.c has a function called DEFUN (defined in lib/command.h: 166-175) for handling this command. It takes a function name, and a command name constructed from the user input value for invoking the appropriate function.

b. Every routing protocol has associated with it a routing table. Whenever a routing protocol wants to fetch routing information from another protocol, it sets the "***redistribute flag***" corresponding to that protocol, in the ***redist*** array. This is done by calling bgp_redistribute_set ( ) (invoked in bgpd/bgp_vty.c: 7092) with the parameters to indicate that the array element corresponding to that protocol has to be set.

c. bgp_redistribute_set( ) (defined in bgpd/bgp_zebra.c: 858-878): In this function, the flag for the particular protocol in the ***redist*** array is set to indicate that routes from the routing table belonging to a particular protocol are to be redistributed into BGP routing table. Now zebra_redistribute_send ( ) is called with parameters specifying the socket in the BGP daemon which is connected to the Zebra daemon, used for fetching information from the Zebra daemon about the routes in OSPF, and the indication that information has to be fetched from the OSPF routing table.

d. zebra_redistribute_send( ) (defined in lib/zclient.c: 510-529): In this function, initially the number of words that are to be written is input to a newly created stream, to inform the protocol daemon about the number of bytes to read from the stream. In the next step, the command ZEBRA_REDISTRIBUTE_ADD is written into the stream which is used to indicate that the operation being performed is addition of redistributed routes into the BGP routing table. The protocol type of routing table (OSPF/RIP/BGP routing information) is also written into the stream.

e. The Zebra daemon which accepts connections from various protocol daemons, receives the above request in the handler for a Zebra service request, zebra_client_read( ) (defined in zebra/zserv.c: 1298-1409, for handling requests such as adding IPv4 routes, redistribution of routes etc). Depending on the request certain functions are invoked. In the case of ZEBRA_REDISTRIBUTE_ADD, the function zebra_redistribute_add( ) is invoked by the Zebra daemon.

f. zebra_redistribute_add( ) (defined in zebra/redistribute.c: 220-246): This function invokes zebra_redistribute( ) with the zclient structure specific to the protocol as a parameter, using which the zebra daemon communicate any requested information.

g. zebra_redistribute( ) (defined in zebra/redistribute.c: 123-146): Before description of the behavior of this function, the two structures of interest are "***struct rib***" and "***struct route_node***". ***rib*** structure is used for storing information of a routing entry in the form of a linked list containing information of all the routing entries. It contains the next hop information, the routing table to which the route belongs to, reference count to this structure, the amount of time for which this route has been valid, metric and the default distance for the route. This structure also holds the information about the number of next hops for a particular node (nexthop_num), the number of active nexthops (nexthop_active_num: used to indicate whether the interface serving the next hop of the route is alive or not). ***route_node*** structure is used for holding the information of all the routing entries in a routing table (for eg., IPv4 and IPv6 entries are maintained in separate routing tables). ***route_node*** structure is used to store the routing information using the radix routing table structure [13]. In each of these entries is a "***void***" pointer which contains a pointer to the associated route information stored in a "***struct rib***" node.

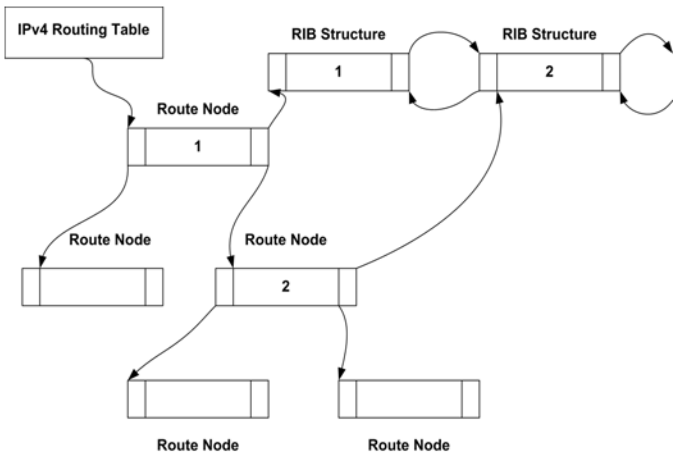


Fig. 19. RIB Structure

Note: *RIB structure 1 contains the route information of the node 1, and RIB structure the route information of the node 2 and so on.*

The RIB structures form a linked list containing information about all the routes in the routing table. The routing entries are present in the form of the route nodes as illustrated in the diagram.

The function zebra_redistribute( ) searches the IPv4 routing table for all the prefixes matching the type of the prefix (IPv4 prefix/IPv6 prefix), and invokes zsend_ipv4_add_multipath( ) with the prefix, RIB element and the zclient as the parameters.

h) zsend_ipv4_add_multipath( ) (defined in zebra/zserv.c: 302-357): This function is used for performing the operation of fetching the route information and writing into the socket associated with the client running the BGP daemon. In this function the RIB type, RIB flags, the prefix, the gateway address associated with the IP address, next hop interface associated with this IP address, the metric associated with this route, are written into the protocol daemon's input socket stream, which effectively completes the process of route redistribution. This marks the crucial point where the actual redistribution occurs and forms a potential candidate for modifying the behavior of redistribution.

   In this function, the output stream associated with the protocol daemon (in this case, BGP), into which the routes are to be redistributed, is identified, and the type of the routing table (IPv4/IPv6 routing table), the route prefix, and the next hop information for the route, the metric associated with the route are written into this output stream. The redistributed routes are installed by the BGP running routers, R1 and R3, into the protocol daemon's routing table.

The route information written into the protocol daemon's socket is installed in the routing table as follows (Please refer to Section 3.2, 3.6 and 3.8 in [11]):

a) The Read thread associated with the protocol daemon always keeps checking the availability of data on the input stream associated with the communication between the protocol and Zebra daemons. On the availability of data to read, zclient_read( ) is invoked.

b) zclient_read( ) (defined in lib/zclient.c: 694-811): The data written into the input stream of the protocol daemon is read in, and in the case of redistributed routes, the Zebra daemon would have indicated the type to be ZEBRA_IPV4_ROUTE_ADD, upon which zclient->ipv4_route_add( ) is invoked. zclient->ipv4_route_add( ) is basically a function pointer, which has been set to point to the function zebra_read_ipv4( ) {in function bgp_zebra_init( )-bgpd/bgp_zebra.c:990}.

c) zebra_read_ipv4( ) (defined in bgpd/bgp_zebra.c: 239-287): This function is used for copying all the information associated with the redistributed route (nexthop information, metric etc.,), and then invokes bgp_redistribute_add( ).

d) bgp_redistribute_add( ) (defined in bgpd/bgp_route.c: 3507-3616): In this function, the redistributed route information is subjected to the policy decisions at the BGP router. In the case wherein the route is denied, all the copied information of the redistributed route is deleted. Otherwise the route is added into the routing table (using bgp_afi_node_get ( ) function defined in bgpd/bgp_route.c:60-87). At the final stage, bgp_process( ) (defined in bgpd/bgp_zebra.c: 717-853) is invoked which is used for advertising the newly added route to the adjacent neighbors. In this way, route redistribution is successfully completed.

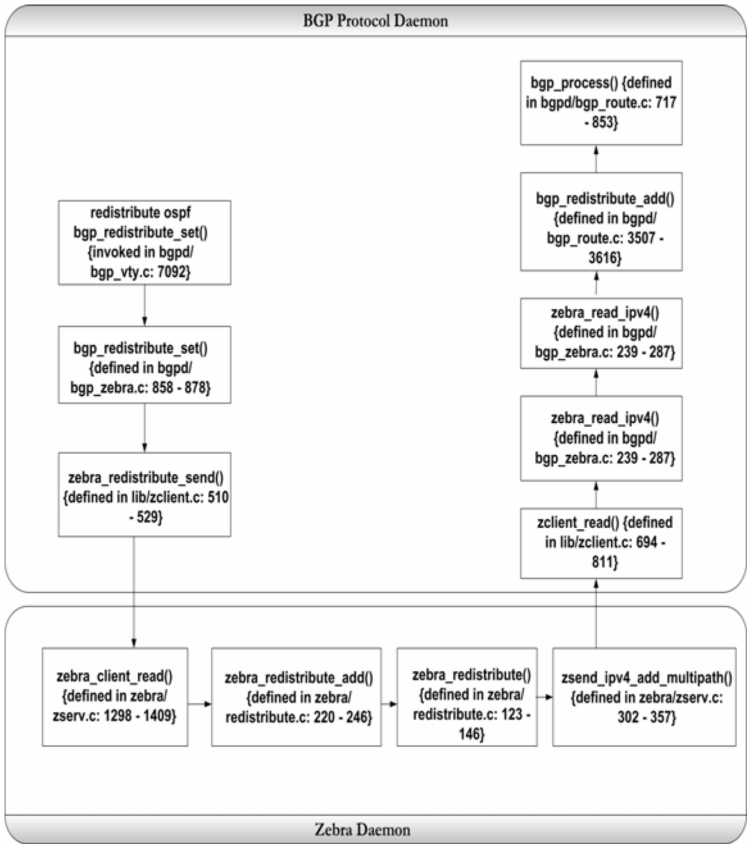

BGP Protocol Daemon
bgp_process() {defined in bgpd/bgp_route.c: 717 - 853}
bgp_redistribute_add() {defined in bgpd/ bgp_route.c: 3507 - 3616}
redistribute ospf bgp_redistribute_set() {invoked in bgpd/ bgp_vty.c: 7092}
zebra_read_ipv4() {defined in bgpd/ bgp_zebra.c: 239 - 287}
bgp_redistribute_set() {defined in bgpd/ bgp_zebra.c: 858 - 878}
zebra_read_ipv4() {defined in bgpd/ bgp_zebra.c: 239 - 287}
zebra_redistribute_send() {defined in lib/zclient.c: 510 - 529}
zclient_read() {defined in lib/zclient.c: 694 - 811}
zebra_client_read() {defined in zebra/ zserv.c: 1298 - 1409}
zebra_redistribute_add() {defined in zebra/ redistribute.c: 220 - 246}
zebra_redistribute() {defined in zebra/ redistribute.c: 123 - 146}
zsend_ipv4_add_multipath() {defined in zebra/zserv.c: 302 - 357}
Zebra Daemon

Fig. 20

Fig. 20 illustrates the flow of control in the source code for BGP and Zebra daemons, for easier understanding.

6. EXPERIMENT METHODOLOGY:

The network topology has been constructed using the VMWare machines, which run as virtual machines with network support included. For the purposes of this experiment, four VMWare machines were set up, and for each of the machines, a virtual network interface was provided. RedHat Linux with linux kernel 2.4.20-20.7 ran on each of the vmware machines. ZebOS Server Routing Suite Standard Edition [7] was used for simulating BGP/OSPF routing between the vmware machines.

BIBLIOGRAPHY

1. Rekhter, Y., Li, T., "An Architecture for IP Address Allocation with CIDR", RFC 1518, September 1993.
2. Fuller, V., Li, T., Yu, J., Varadhan, K., "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", September 1993.

ACRONYMS

1. BGP – Border Gateway Protocol
2. OSPF – Open Shortest Path First
3. CIDR – Classless Inter-Domain Routing
4. RIP – Routing Information Protocol
5. AS – Autonomous System
6. RIB – Routing Information Base
7. FIB – Forwarding Information Base

REFERENCES

1. Rekhter, Y., Li, T., "A Border Gateway Protocol 4 (BGP – 4)", RFC 1771, March 1995.
2. Hedrick, C., "Routing Information Protocol", RFC 1058, June 1998.
3. Malkin, G., "RIP Version 2", RFC 2453, November 1998.
4. Moy, J., "OSPF Version 2", RFC 2328, March 1994.
5. GateD® Enterprise 2.0, NextHop Technologies, http://www.nexthop.com/products/gated.shtml.
6. GNU Zebra, http://www.zebra.org/.
7. ZebOS Server Routing Suite Standard Edition, http://www.ipinfusion.com/products/server/standard edition.html.
8. Malkin, G., Minnear, R., "RIPng for IPv6", January 1997.
9. Coltun, R., Ferguson, D., Moy, J., "OSPF for IPv6", December 1999.
10. Quagga Routing Software Suite, http://www.quagga.net.
11. Michael Feng, Roy Leung, Andrew Do-Sung Jun, Fig 3.1, "Summer Report 1999, Linux Network", Electrical and Computer Engineering, University of Toronto, http://dcn.ssu.ac.kr/~softgear/prog_sw_report_summer_99.pdf
12. Gary R. Wrights, W. Richard Stevens, "TCP/IP Illustrated",Volume 2, Chapter 18.